Computer Graphics

Tom Ellman
Lecture 10
How to Think about the Modelview Matrix?

```cpp
glTranslatef(1.0, 1.0, 1.0);
glRotatef(-90.0, 1.0, 0.0, 0.0);
glScalef(1.0, 0.5, 1.0);
glutWireCube(1.0);
```
Two Ways of Thinking

• Specifying operations to perform on an object after it is created, but before it is drawn.

• Specifying operations that define a local coordinate system for drawing the object.
Operating on the Object

- Operations are performed on the object in the reverse of the order in which they are executed in the program code.
- In code:
  1. glTranslate; 2. glRotate; 3. glScale.
- On object:
  1. Scale; 2. Rotate; 3. Translate.
Scaling the Object after it is Created
Rotating the Object after it is Created
Translating the Object after it is Created
Defining a Local Coordinate System

- Each call to `glTranslate`, `glRotate` or `glScale` defines a new local coordinate system.
- The new coordinate system is translated, rotated or scaled relative to the existing system.
- Changes to the coordinate system happen in the same order as `glTranslate`, `glRotate` and `glScale` appear in the program code.
Translating the Coordinate System
Rotating the Coordinate System
Scaling the Coordinate System
Drawing the Unit Cube in the Local Coordinate System
Change of Coordinates  
(Fix Origin; Change Basis)

\[ x' = Ax \]
\[ y' = Ay \]
If $A$ transforms the basis then $A^{-1}$ transforms the coordinates of a point.

- Let $A$ be an invertible linear transformation that converts basis $\{x_i \mid i = 1 \ldots n\}$ into basis $\{x'_i \mid i = 1 \ldots n\}$.
- Each new basis vector results from applying $A$ to one old basis vector: $x'_i = Ax_i$, (for $i = 1 \ldots n$).
- Let $v$ be a column vector of coordinates of a point in terms of the old basis.
- To represent the vector $v$ in terms of the new basis, we solve the equation $Av' = v$.
- The solution is: $v' = A^{-1}v$.
- If $A$ transforms the basis, then $A^{-1}$ is the corresponding change in coordinates.
Change of Coordinates
(Fix Basis; Change Origin)

\[
\begin{pmatrix}
X' \\
Y'
\end{pmatrix}
= \begin{pmatrix}
X \\
Y
\end{pmatrix} - t
\]

\[
o' = o + t
\]
If $A$ translates the origin then $A^{-1}$ transforms the coordinates of a point.

- Let $A$ be a translation that moves a vector $v$ to $v + t$.
- Then $A$ may be used to define a translation of the origin of a coordinate system: $o' = o + t = Ao$.
- Let $v$ be a column vector of the homogeneous coordinates of a point $P$ in the original coordinate system.
- Let $v'$ be a column vector of the homogeneous coordinates of point $P$ in the translated coordinate system.
- Then $v' = v - t = A^{-1}v$. 
Summary of Transformation Rules

- Transformation $A$ converts the world coordinate system into a local coordinate system.
- Transformation $A^{-1}$ converts world coordinates into local coordinates.
- Transformation $A$ converts local coordinates back into world coordinates.
Modeling a Robot Arm

• The arm has three parts:
  – An upper arm.
  – A forearm.
  – A hand.

• The arm has three joints.
  – A shoulder.
  – An elbow.
  – A wrist.
void drawRobotArm()
{
    glPushMatrix();
gotoShoulderCoordinates();
drawUpperArm();
gotoElbowCoordinates();
drawForeArm();
gotoWristCoordinates();
drawHand();
glPopMatrix();
}
Setting up a Local Coordinate System at the Shoulder Joint

- Define a coordinate system.
- Centered at the global origin.
- Whose X, Y and Z axies have rotated along with the upper arm.
Setting up a Local Coordinate System at the Shoulder
Setting up a Local Coordinate System at the Shoulder Joint

```c
void gotoShoulderCoordinates()
{
    glRotatef ( shoulderX, 1.0, 0.0, 0.0);
    glRotatef ( shoulderY, 0.0, 1.0, 0.0);
    glRotatef ( shoulderZ, 0.0, 0.0, 1.0);
}
```
void drawUpperArm()
{
    glPushMatrix();
    glColor3f (1.0, 0.0, 0.0);
    glRotatef(90.0, 0.0, 1.0, 0.0);
    gluCylinder(upperArm,0.1,0.1,1.0,8,1);
    glPopMatrix();
}
Setting up a Local Coordinate System at the Elbow

• Define a coordinate system.
• Centered at the elbow.
• Whose X, Y and Z axies have rotated (relative to the upper arm) along with the forearm.
• Since the shoulder-centered coordinate system has rotated along with the upper arm, we move to the elbow with an X axis parallel translation.
Setting up a Local Coordinate System at the Elbow
void gotoElbowCoordinates() {
    glTranslatef (1.0, 0.0, 0.0);
    glRotatef (elbowX, 1.0, 0.0, 0.0);
    glRotatef (elbowY, 0.0, 1.0, 0.0);
    glRotatef (elbowZ, 0.0, 0.0, 1.0);
}
void drawForeArm()
{
    glPushMatrix();
    glColor3f(0.0, 1.0, 0.0);
    glRotatef(90.0, 0.0, 1.0, 0.0);
    gluCylinder(foreArm, 0.1, 0.1, 1.0, 8, 1);
    glPopMatrix();
}
Setting up a Local Coordinate System at the Wrist

- Define a coordinate system.
- Centered at the wrist.
- Whose X, Y and Z axes have rotated (relative to the forearm) along with the hand.
- Since the elbow-centered coordinate system has rotated along with the forearm, we move to the wrist with an X axis parallel translation.
Setting up a Local Coordinate System at the Wrist
Setting up a Local Coordinate System at the Wrist

```c
void gotoWristCoordinates()
{
    glTranslatef (1.0, 0.0, 0.0);
    glRotatef (wristX, 1.0, 0.0, 0.0);
    glRotatef (wristY, 0.0, 1.0, 0.0);
    glRotatef (wristZ, 0.0, 0.0, 1.0);
}
```
void drawHand()
{
    glPushMatrix();
    glColor3f (1.0, 1.0, 0.0);
    glRotatef(90.0, 0.0, 1.0, 0.0);
    gluCylinder(hand,0.1,0.1,1.0,8,1);
    glPopMatrix();
}
Robot Arm Scene Graph

Diagram of a robot arm scene graph with nodes for upperArm, foreArm, hand, and TransformNode connections.
Modeling a Solar System

• The system has a sun, an earth and two moons.
  – The two moons revolve around the earth.
  – The whole earth system revolves around the sun.
  – Each object rotates on its own axis.

• We display each body as a sphere.

• We also display each body’s orbit as a circle.
Drawing the Solar System

```java
void drawSolarSystem()
{
    drawSun();
    drawEarthOrbit();
    drawEarthSystem();
}
```
void drawSun()
{
    glPushMatrix();
    glColor3f (1.0, 1.0, 0.0);
    glRotatef(sunRotation, 0.0, 0.0, 1.0);
    glutWireSphere(SUN_RADIUS, 20, 16);
    glPopMatrix();
}
void drawEarthOrbit()
{
    glColor3f (0.0, 0.0, 1.0);
    drawCircle(EARTH_ORBIT_RADIUS);
}

Drawing the Earth’s Orbit
void drawEarthSystem() {
    glPushMatrix();
    gotoEarthCoordinates();
    drawEarth();
    drawMoon1Orbit();
    drawMoon1System();
    drawMoon2Orbit();
    drawMoon2System();
    glPopMatrix();
}
void gotoEarthCoordinates()
{
    glRotatef (earthRevolution, 0.0, 0.0, 1.0);
    glTranslatef (EARTH_ORBIT_RADIUS, 0.0, 0.0);
    glRotatef (-earthRevolution, 0.0, 0.0, 1.0);
}
void drawEarth()
{
    glPushMatrix();
    glRotatef (earthRotation, 0.0, 0.0, 1.0);
    glutWireSphere(EARTH_RADIUS, 10, 8);
    glPopMatrix();
}

Drawing the Earth
Drawing the Moons’ Orbits

void drawMoon1Orbit() {
    glColor3f (0.0, 0.0, 1.0);
    drawCircle(MOON1_ORBIT_RADIUS);
}

void drawMoon2Orbit() {
    glColor3f (0.0, 0.0, 1.0);
    drawCircle(MOON2_ORBIT_RADIUS);
}
void drawMoon1System()
{
    glPushMatrix();
goToMoon1Coordinates();
drawMoon1();
glPopMatrix();
}

void drawMoon2System()
{
    glPushMatrix();
goToMoon2Coordinates();
drawMoon2();
glPopMatrix();
}
Setting Up Moon-Centered Coordinate Systems

```c
void gotoMoon1Coordinates()
{
    glRotatef (moon1Revolution, 0.0, 0.0, 1.0);
    glTranslatef (MOON1_ORBIT_RADIUS, 0.0, 0.0);
    glRotatef (-moon1Revolution, 0.0, 0.0, 1.0);
}

void gotoMoon2Coordinates()
{
    glRotatef (moon2Revolution, 0.0, 0.0, 1.0);
    glTranslatef (MOON2_ORBIT_RADIUS, 0.0, 0.0);
    glRotatef (-moon2Revolution, 0.0, 0.0, 1.0);
}
```
void drawMoon1()
{
    glRotatef (-moon1Rotation, 0.0, 0.0, 1.0);
    glutWireSphere(MOON1_RADIUS, 10, 8);
}

void drawMoon2()
{
    glRotatef (-moon2Rotation, 0.0, 0.0, 1.0);
    glutWireSphere(MOON2_RADIUS, 10, 8);
}
Solar System Scene Graph

Root

solarSystemXfn :TransformNode

sunXfn :TransformNode

sun :ShapeNode

earthOrbitXfn :TransformNode

earthOrbit :ShapeNode

earthSystemXfn :TransformNode

earthXfn :TransformNode

earth :ShapeNode

m1OrbitXfn :TransformNode

m1Orbit :ShapeNode

m1SystemXfn :TransformNode

m1Xfn :TransformNode

m1 :ShapeNode

m2OrbitXfn :TransformNode

m2Orbit :ShapeNode

m2SystemXfn :TransformNode

m2Xfn :TransformNode

m2 :ShapeNode